UNO Game Programming Assignment

Programming Assignment 4

Overview

The purpose of programming assignment 4 is to create a simulation of the card game UNO that allows the efficacy of various card selection strategies to be studied. To this end your job is to supply a Player class that implements the strategy of your choice for selecting an UNO card to play. The code for a Player object must follow the rules of UNO as published in the official rules of the game. (http://unotips.org/pdf/official_rules.pdf). *Read the official rules. They are different than those used by most people, and the simulation follows official the rules!*

Resources you will need are included in the GitHub repo of the project and online. The files you need are:

- UNOGameAssignment.pdf (this document) - The details of the assignment, including the design of the simulation application. You will find a copy in the /doc folder of the GitHub repo. **Start by reading through this document.**
- CSLibrary.jar - This library archive contains the CSLibrary code. The jar file is included in the GitHub repository. You will need this library in order for the Deck and Hand classes in the cards package to work. ***Very Important: Link your project code to the CSLibrary that is provided!***
- TestPlan.pdf - This document describes the tests provided in the test folder of the project. It is also located in the /doc folder of the repository. **Read this document after you have read the assignment document.**
- official_rules.pdf. The official rules of UNO. This document serves as the definitive requirements for the game and player. A copy is in the /doc folder of the GitHub repository.
- The API documentation for the CSLibrary classes can be found at: http://mcs.drury.edu/ssigman/CSLibrary/
- The API documentation for the UNO project packages and classes can be found at: http://mcs.drury.edu/UNO/
- Starter code for the project is in the GitHub repo provided to you.

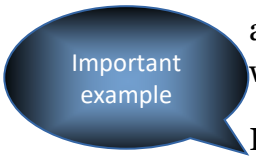
Your Assignment

Your job in this assignment is to write a player class that implements some strategy for playing an UNO card when it is your turn to play. Requirements can be found in the documentation provided above.

Work this way! (Important!!)

After you have read through the documentation and have an idea of how the simulation is designed, read through the test plan. Construct your Player class by writing a player that can pass the BasicPlayTests. After your code can pass the BasicPlayTests, *commit the code to your repo with the commit message "Basic Play Tests Complete",* choose another set of tests and modify your code to pass these tests. The modified code should still pass the BasicPlayTests. Continue in this fashion until you have a Player class that will pass all tests. When you do, commit and push your code with the commit message "Test plan complete." We will integrate your code into the full simulation and see if it works.

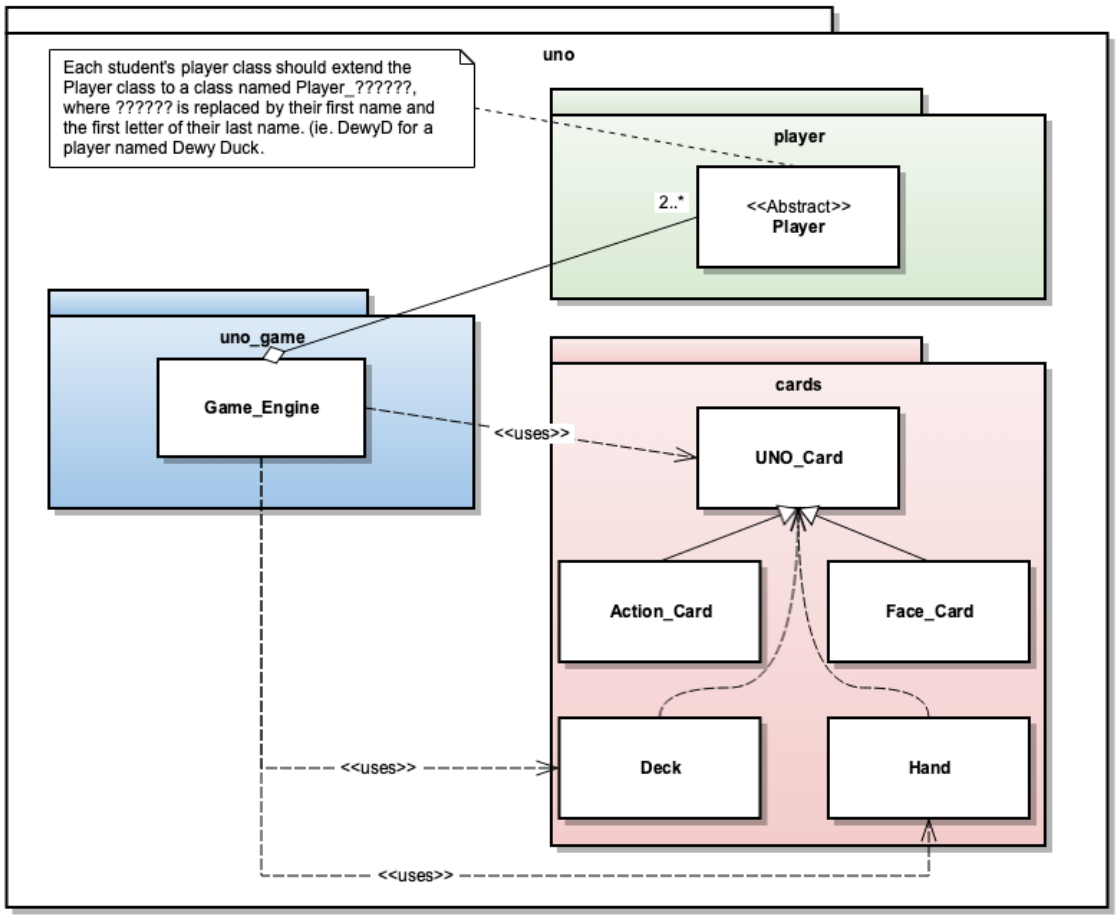
The CSLibrary library provides implementations for several collection classes: Stack, a BasicList, UnorderedLists, and an OrderedLists. You may use any of these collections in the construction of your Player class. You may also use any of the collections available in the standard Java Libraries. (Remember, Java library classes are documented in the Java API.) The choice is yours and as long as you abide by the specified design for your Player class, there should be no problem with either choice.



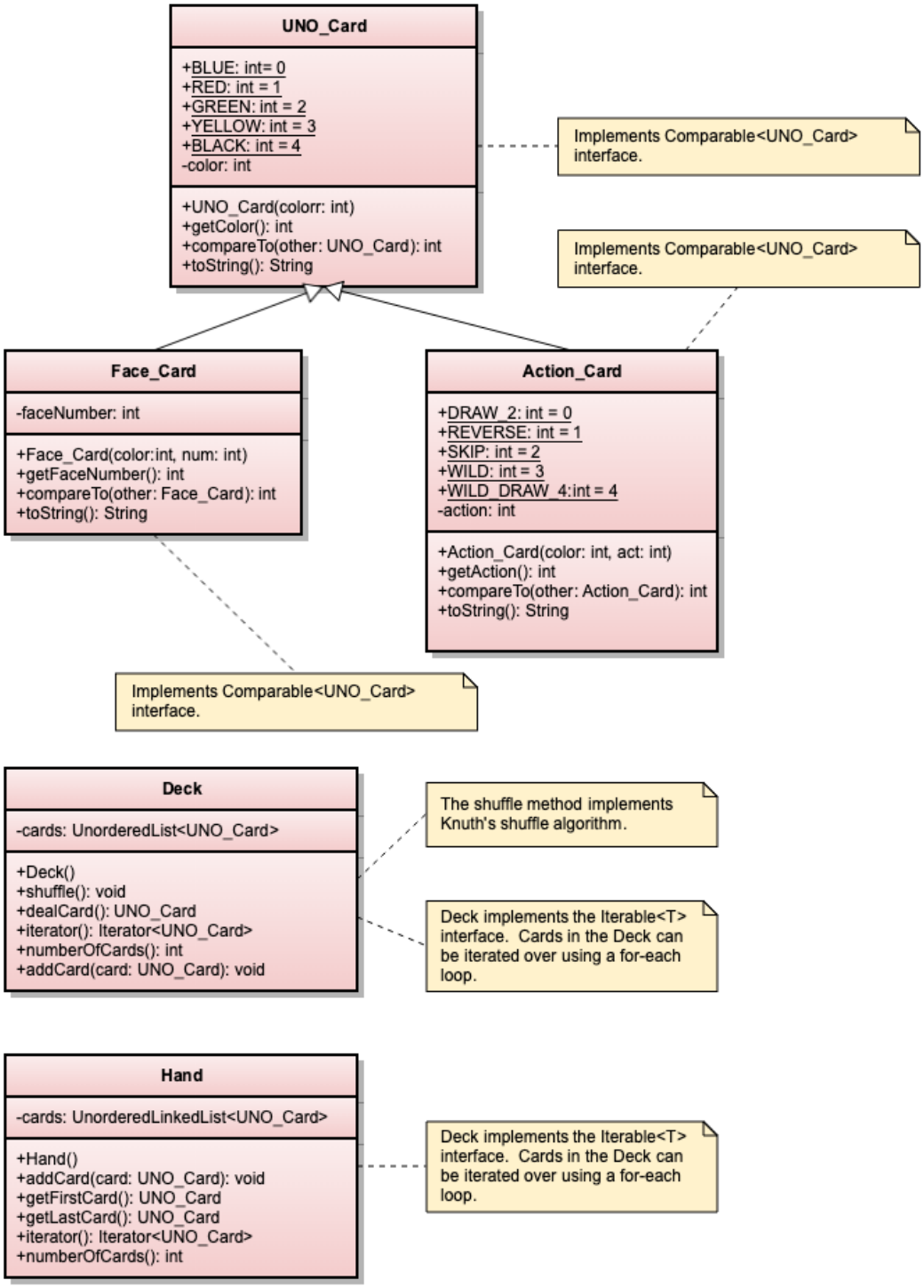
For example, you might use a simple queue to implement a strategy to always play the first legally playable card in your hand. Your Player would remove cards from the front of the queue and check to see if they were playable. If not, they would be added back to the end of the queue. This would continue until a playable card was found or all cards had been tried. More sophisticated strategies may require different data structures.

The portion of the system that controls game play will be implemented by the instructor. The design of the system is specified in the following sections.

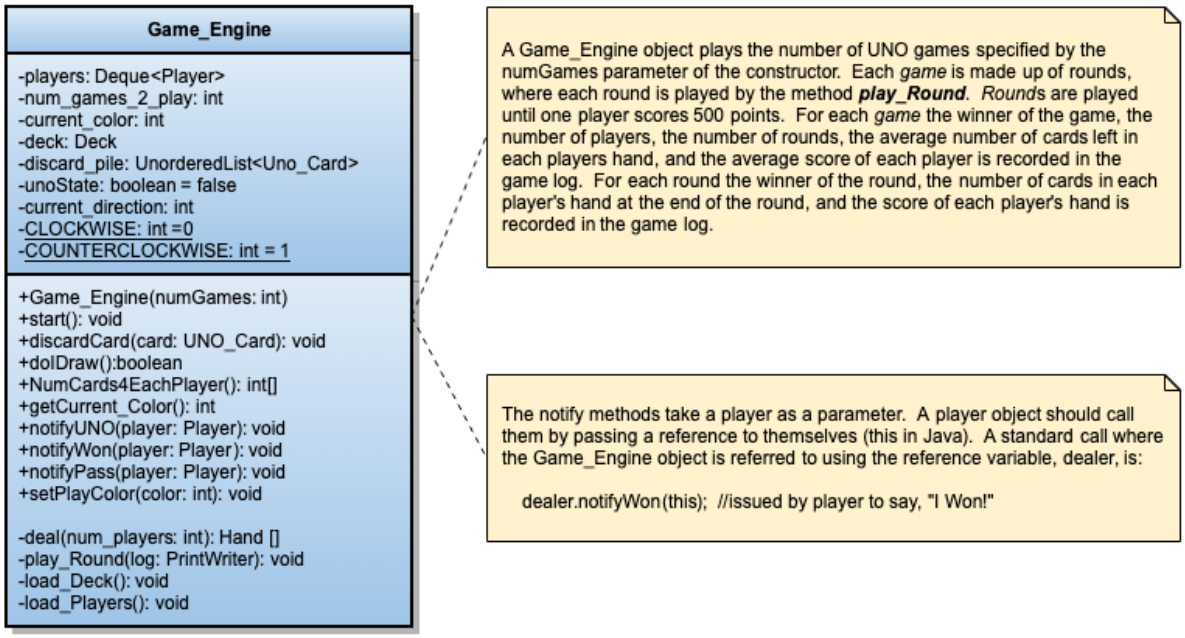
Package Design



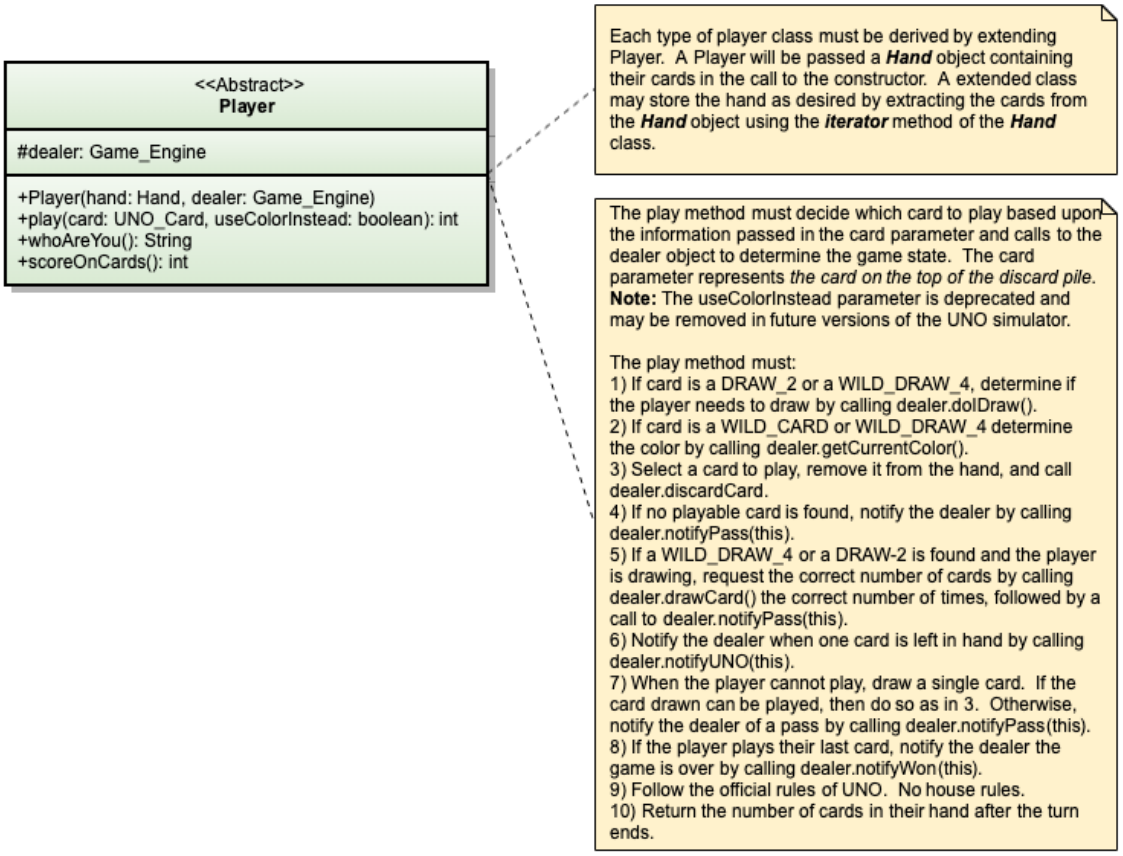
Design of the Card Package



Design of the UNO Game Package



Design of the Player Package



List Design

The instructor's implementation of the card and game package class depend upon the following list design. Your list package must conform for the instructor's code to work correctly.